Оглавление.

Цель эксперимента.

Найти наиболее выгодный метод работы с динамической памятью при обработке данных, объем которых заранее неизвестен.

Постановка задачи.

Задан файл с некоторым заранее неизвестным количеством целых чисел. Необходимо считать числа в динамический массив.

Описание методов.

Метод №1 (метод подсчета).

При использовании данного метода файл будет прочитан дважды. При первом проходе по файлу считается количество чисел в нем. Затем выделяется необходимое количество памяти и во время второго прохода данные сохраняются в массив.

Преимущества метода:

1) Относительная простота работы с динамической памятью.

Недостатки метода:

1) Необходимость читать файл дважды.

Метод №2 (метод realloc-а №1).

При использовании данного метода файл читается единожды. При необходимости дополнительной памяти массив динамически расширяется с запасом. Новый массив получается в два раза больше исходного за счет увеличение в два раза от старого размера.

Преимущества метода:

1) Файл читается один раз.

Недостатки метода:

1) Сложности при использование realloc, необходимость следить за использованием динамической памяти и правильным расширением.
2) В худшем случае копирование данных из одной области памяти в другую.

Метод №3 (метод realloc-а №2).

При использовании данного метода файл читается единожды. При необходимости дополнительной памяти массив динамически расширяется. Новый массив получается на один элемент больше исходного за счет увеличение на единицу от старого размера.

Преимущества метода:

1) Файл читается один раз.

Недостатки метода:

1) Сложности при использование realloc, необходимость следить за использованием динамической памяти и правильным расширением.
2) В худшем случае копирование данных из одной области памяти в другую.

Листинги программ.

Метод подсчета.

```
int alloc_mem_with_count(FILE *f, int **arr, size_t *n_mem) {
    int rc = 0;
    size_t count = 0;
    while (rc == 0) {
        int tmp;
        if (fscanf(f, "%d", &tmp) != 1)
            rc = 1;
        else
            count++;
    }
    if (feof(f))
        rc = 0;
    if (count <= 0)
        rc = 4;
    rewind(f);
    if (rc == 0) {
        *arr = malloc(count * sizeof(int));
        if (!*arr)
            rc = 2;
        else {
            for (size_t i = 0; i < count; ++i)
                fscanf(f, "%d", &(*arr)[i]);
        }
    }
    *n_mem = count;
    return rc;
}
```

Метод realloc-а №1.

```
int alloc_mem_with_capacity_1(FILE *f, int **arr, size_t *n_mem)
{
    *arr = NULL;
    size_t count = 0;
    size_t capacity = 0;
    int rc = 0;
    while (rc == 0) {
        int tmp;
        if (fscanf(f, "%d", &tmp) != 1)
            rc = 1;
        else {
```

```
            if (count ⩾ capacity) {
                capacity = capacity == 0 ? 1 : capacity * 2;
                int *new_tmp_arr = realloc(*arr, capacity *
sizeof(int));
                if (new_tmp_arr == NULL) {
                    rc = 2;
                    free(*arr);
                } else {
                    *arr = new_tmp_arr;
                }
            }
            (*arr)[count++] = tmp;
        }
    }
    if (rc == 1 && feof(f)) {
        rc = 0;
    }
    *n_mem = count;
    return rc;
}
```

Метод realloc-а №2.

```
int alloc_mem_with_capacity_2(FILE *f, int **arr, size_t *n_mem)
{
    *arr = NULL;
    size_t count = 0;
    size_t capacity = 0;
    int rc = 0;
    while (rc == 0) {
        int tmp;
        if (fscanf(f, "%d", &tmp) ≠ 1)
            rc = 1;
        else {
            if (count ⩾ capacity) {
                capacity++;
                int *new_tmp_arr = realloc(*arr, capacity *
sizeof(int));
                if (new_tmp_arr == NULL) {
                    rc = 2;
                    free(*arr);
                } else {
                    *arr = new_tmp_arr;
                }
            }
```

```
            (*arr)[count++] = tmp;
        }
    }
    if (rc == 1 && feof(f)) {
        rc = 0;
    }
    *n_mem = count;
    return rc;
}
```

Обработка результатов.

Результаты замеров.

	Метод подсчета			Метод realloc-а №1			Метод realloc-а №2		
Кол-во данных	t, мс	Кол-во замеров	RSE, %	t, мс	Кол-во замеров	RSE, %	t, мс	Кол-во замеров	RSE, %
0	0,017	2000	0,89	0,017	2000	0,97	0,018	2000	0,99
1	0,022	2000	0,96	0,016	2000	0,86	0,017	2000	0,91
10	0,023	2000	0,95	0,020	2000	0,94	0,020	2500	0,91
50	0,027	2000	0,96	0,023	2000	0,94	0,022	2000	0,96
100	0,034	2000	0,97	0,027	2000	0,97	0,028	2500	0,90
500	0,085	2000	0,96	0,054	2500	0,89	0,057	2000	0,96
1000	0,15	2000	0,91	0,087	2000	0,95	0,094	2000	0,92
1500	0,19	2000	0,76	0,12	2000	0,93	0,12	2000	0,82
2000	0,28	2000	0,94	0,15	2000	0,93	0,17	2000	0,90
2500	0,31	2000	0,80	0,18	2000	0,90	0,22	2500	0,90
3000	0,38	2000	0,84	0,24	2000	0,94	0,25	2000	0,94
3500	0,43	2000	0,74	0,26	2000	0,97	0,31	2500	0,91
4000	0,52	2000	0,80	0,28	2000	0,88	0,35	2500	0,89
4500	0,58	2000	0,77	0,31	2000	0,90	0,37	2000	0,97
5000	0,68	2000	0,80	0,33	2000	0,85	0,38	2000	0,79
5500	0,71	2000	0,73	0,36	2000	0,83	0,42	2000	0,88
6000	0,80	2000	0,72	0,41	2000	0,87	0,48	2000	0,95
6500	0,82	2000	0,68	0,46	2000	0,90	0,50	2000	0,88
7000	0,91	2000	0,64	0,45	2500	0,70	0,59	2000	0,97
7500	0,93	2000	0,63	0,53	2000	0,85	0,59	2000	0,91
8000	0,94	2000	0,60	0,55	2000	0,83	0,62	2000	0,89

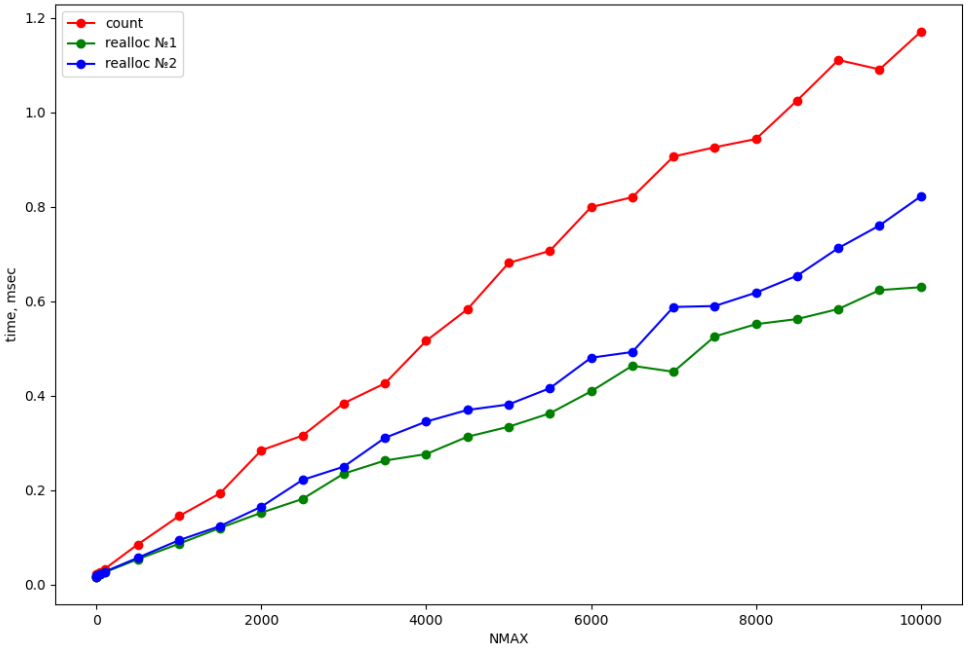
8500	1,0	2000	0,58	0,56	2000	0,80	0,65	2000	0,89
9000	1,1	2000	0,62	0,58	2000	0,76	0,71	2000	0,90
9500	1,1	2000	0,52	0,62	2000	0,74	0,76	2000	0,91
10000	1,2	2000	0,55	0,63	2000	0,69	0,82	2000	0,93

Графическое представление данных.

count (красный) - метод подсчета.

realloc №1 (зеленый) - метод realloc-а №1 (увеличение в два раза).

realloc №2 (синий) - метод realloc-а №2 (увеличение на единицу).



Вывод.

По полученным данным видно, что метод с использованием realloc-а при увеличении массива с запасом выигрывает в два раза по сравнению с методом подсчета. Также данный метод выигрывает по времени над методом с использованием realloc-а с увеличением на единицу. Метод подсчета проигрывает обеим реализациям с использованием realloc-а. В связи с этим можно сделать вывод, что несмотря на сложность работы с realloc-ом он дает преимущество по времени по сравнению с методом подсчета, даже учитывая, что в худшем случае использование функции realloc вызывает полное копирование данных из одной области памяти в другую. Увеличение массива с запасом дает наибольший выигрыш по времени из рассмотренных реализаций, хоть и требует использование большего количества памяти.